MODUL 4. FILTER STREAM

4.1 *Kompetensi*

- Memahami fungsi dan mengetahui kelebihan dari filter stream
- Menerapkannya dalam aplikasi

4.2 *Alat dan Bahan*

- OS Windows
- OS Linux (Ubuntu)
- Netbeans

4.3 *Ulasan Teori*

Karena stream dasar hanya menerapkan mekanisme pembacaan byte dari informas atau data secara sederhana, maka fleksibilitasnya terbatas. Pembacaan byte dari data tidaklah simpel, arti dari byte bisa bervariasi tergantung jenis datanya. Teks contohnya, merupakan rangkaian dari karakter karakter, dan contoh lain adalah deretan angka-angka yang mana masing-masing karakter dan angka bisa dibentuk lebih dari satu byte data tunggal. Komunikasi pada level byte juga bisa sangat tidak efisien, dan pemberian buffer juga dapat meningkatkan performasi dari sistem. Permasalahan-permasalahan tersebut bisa diatasi dengan penggunaan filter stream.

4.3.1 *Dasar Filter Stream*

Filter Stream menambahkan fungsionalitas pada stream-stream dasar, dengan memproses data sedemikian rupa (seperti penambahan buffer untuk meningkatkan kinerja) atau menambahkan method-method untuk mengakses data dalam cara yang berbeda (seperti membaca data per baris, dibandingkan pembacaan data standar yang byte per byte).

Filter Stream dibagi pula menjadi 2 kategori : filter input stream dan filter output stream. Setiap filter stream disambungkan ke input stream atau output stream dalam penggunaannya. Menyambungkan filter stream sangat sederhana, buat objek filter streamnya (dengan memasukkan stream yang sudah ada ke parameter input method pembentuknya), dan selanjutnya proses pembacaan atau penulisan data cukup melalui filter stream tersebut.

Contoh apabila kita ingin menggunakan salah satu filter output stream yang bertipe PrintStream (digunakan untuk mencetak teks ke subclass dari Output Stream) pada stream output yang menuliskan data ke file. Kode di bawah ini dapat digunakan :

```
FileOutputStream fout = new FileOutputStream ( somefile );
PrintStream pout = new PrintStream (fout);
pout.println ("hello world");
```

Dapat dilihat bahwa penulisan atau pembacaan yang tadinya dilakukan pada stream dasar, apabila telah dihubungkan dengan sebuah filter stream, maka penulisan atau pembacaan data dilakukan pada filter streamnya.

4.3.2 *Filter Input Stream*

Beberapa filter input stream yang umum dan terdapat pada semua versi Java dalah sebagai berikut :

Tabel 4.1 Filter Input Stream

Filter Input Stream	Keguanaan
BufferedInputStream	Memberi buffer pengaksesan data, untuk meningkatkan efisiensi dan kinerja
DataInputStream	Membaca tipe data primitif (int, float, double etc)
LineNumberInputStream	Menjaga hitungan jumlah baris dari data yang sedang dibaca berdasarkan karakter-karakter yang memiliki interpretasi end-of-line
PushBackInputStream	Filter yang memungkinkan pembacaan data dan mengembalikannya lagi ke dalam stream untuk dibaca lagi.

4.3.3 *Buffered Input Stream*

Kegunaan dari memberikan buffer pada I/O adalah menambah performasi kerja. Dibandingkan dengan pembacaan data byte per byte, sejumlah besar byte dibaca bersamaan pada saat method read()class ini dieksekusi. Hal ini meningkatkan kecepatan waktu akses dari sumber data dan mengurangi jumlah blok aplikasi ke input yang menjadi sumber data.

- **Konstruktor buffered input stream :**

- *BufferedInputStream(inputStream input)* : membuat stream dengan buffer yang akan membaca dari input stream yang dijadikan parameter
- *BufferedInputStream(InputStream input, int bufferSize)* : membuat stream dengan buffer sesuai ukuran yang ditentukan, yang akan membaca dari input stream yang dijadikan parameter.

Method-method :

Tidak ada method tambahan. Hanya memodifikasi method read(), dan mendukung pemakaian method mark() dan reset().

4.3.4 *Data Input Stream*

Penulisan dan pembacaan data bertipe primitif seperti angka dan karakter sangat sering dijumpai dalam pemrograman. Informasi / data seperti itu tidak mudah direpresentasikan dalam byte-byte. Dengan filter stream ini, pengembang aplikasi tidak perlu susah-susah dalam mengkonversi data byte ke data primitif. Tetapi cukup dengan memanfaatkan method dari kelas filter stream ini, yang otomatis melakukan mentranlasi data.

- **Konstruktor :**
  - *DataInputStream(InputStream input)*
- **Method-method :**
  - *Boolean readBoolean( )*
  - *Byte readByte( )*
  - *char readChar( )*
  - *double readDouble( )*
  - *float readFloat( )*
  - *void readFully(byte[ ] byteArray)*
  - *void readFully(byte[] byteArray, int offset, int length)*
  - *float readInt( )*
  - *string readLine( )*
  - *long readLong( )*
  - *short readShort( )*
  - *int readUnsignedByte( )*
  - *int readUnsignedShort( )*
  - *String readUTF( )*
  - *Static String readUTF(DataInputStream input)*

- *int skipBytes(int number).*

4.3.5 *LineNumber Input Stream*

Memberikan fungsionalitas yang berguna yaitu memantau jumlah baris data yang telah dibaca dari input stream.

Konstruktor :

- *LineNumberInputStream(InputStream input)* – membuat sebuah line number stream, yang membaca dari input stream *input*

Methods

- *int getLineNumber( )* – mengembalikan nilai jumlah baris yang telah dibaca dari input stream yang bersangkutan
- *void setLineNumber(int number)* – mengubah nilai *counter* nomor baris menjadi nilai yang ditentukan dalam *number*

4.3.6 PushBackInputStream

Memiliki kemampuan untuk membaca sebuah byte data tunggal, lalu dikembalikan lagi ke dalam input stream untuk dibaca kemudian. Untuk itu dia mempunyai sebuah buffer internal. Kelas ini berguna bila programmer ingin 'mengintip' data apa yang akan dibaca selanjutnya dari rangkaian byte data pada input stream.

- **Konstruktor**
  - *PushBackInputStream(InputStream input)*
  - *PushBackInputStream(InputStream input, int bufferSize)* – membuat PushBackInputStream yang akan membaca dari input stream dan menggunakan buffer dengan besar yang ditentukan
- **Method-method**
  - *void unread(byte[] byteArray)* – mengembalikan isi dari array *byteArray*.
  - *void unread(byte[] byteArray, int offset, int length)*
  - ***void unread(int byte)*** – mengembalikan byte yang diinputkan ke awal dari buffer

4.3.7 *Filter Output Stream*

Beberapa filter output stream yang umum dan terdapat pada semua versi Java adalah sebagai berikut :

Tabel 4.2 Kegunaan Output Stream

Filter Output Stream	Kegunaan
BufferedOutputStream	Menyediakan buffer untuk menuliskan data
DataOutputStream	Menuliskan tipe data primitif seperti bytes dan angka.
PrintStream	Menyediakan method tambahan untuk menuliskan baris teks atau tipe data lain sebagai sebuah teks

4.3.8 *Buffered Output Stream*

Memiliki buffer internal yang dikelolanya, dimana apabila buffer tersebut penuh, maka isi dari buffer akan dikosongkan untuk dimasukkan ke output stream dimana dia terhubung.

- **Konstruktor**
  - *BufferedOutputStream(OutputStream output)* – membuat buffer untuk menuliskan data pada output stream yang ditunjuk (default buffer nya sebesar 512 bytes)
  - *BufferedOutputStream(OutputStream output, int bufferSize)* – membuat objek bufferedOutputStream yang terhubung pada output stream yang ditunjuk dengan ukuran buffer yang ditentukan.
- **Methods**

Tidak memiliki method tambahan selain bawaan dari kelas Output Stream.

4.3.9 *Data Output Stream*

Didesain untuk bekerja dengan tipe data primitif seperti angka-angka atau bytes.

- **Konstruktor**
  - *DataOutputStream(OutputStream output)* – membuat objek data output stream yang terhubung dengan output stream yang ditunjuk

- **Method**

- *int size( )* – mengembalikan jumlah byte yang telah dikirimkan pada data output stream
- *void writeBoolean(boolean value)* – menuliskan nilai boolean yang ditentukan dalam bentuk byte
- *void writeByte(int byte)* – menuliskan byte yang ditentukan pada output stream
- *void writeBytes(String string)* – menuliskan seluruh isi string ke output stream sebagai byte, satu per satu byte
- *void writeChar(int char)* – menuliskan karakter ke output stream sebagai nilai byte (2 byte per karakter)
- *void writeChars(String string)* – menuliskan seluruh isi string ke output stream sebagai byte, masing-masing karakter diwakili 2 byte
- *void writeDouble(double doubleValue)* – mengkonversi nilai data *double* yang dimasukkan menjadi nilai *long*, lalu mengkonversinya lagi ke dalam bentuk 8-byte data.
- *void writeFloat(float floatValue)* – mengkonversi nilai data *float* yang dimasukkan menjadi nilai *int*, lalu mengkonversinya lagi ke dalam bentuk 4-byte data.
- *void writeInt(int intValue)* – menuliskan nilai *int* sebagai 4-byte data
- *void writeLong(int intValue)* – menuliskan nilai *long* sebagai 8-byte data
- *void writeShort(int intValue)* – menuliskan nilai *short* sebagai 2-byte data
- *void writeUTF(String string)* – menuliskan isi string menggunakan UTF-8 encoding

4.3.10 *PrintStream*

Kelas ini menyediakan cara yang mudah untuk menuliskan tipe data primitif sebagai bentuk teks menggunakan method print(..) atau println (..) untuk menuliskannya dengan memberikan *line separator*.

- **Konstuktor** :
  - *PrintStream(OutputStream output)*

- *PrintStream(OutputStream output, boolean flush)* – membuat print stream dimana bila nilai boolean diset '*true*', maka buffer yang dimilikinya bisa otomatis melakukan '*flushing*' data

- **Method**
  - *boolean checkError( )* – automatically flushed the output stream and checks to see if an error has occurred
  - *void print(boolean value)* – print a boolean value
  - *void print(char character)* – prints a character value
  - *void print(char [] charArray)* – prints an array of characters
  - *void print(double doubleValue)* – prints a double value
  - *void print(float floatValue)* – prints a float value
  - *void print(int intValue)* – prints an int value
  - *void print(long longValue)* – prints a long value
  - *void print(Object obj)* – prints the value of the specified object's *toString( )* method
  - *void print(String string)* – prints a string's contents
  - *void println( )* – sends a line separator (\n)
  - *void println(char character)* – prints a character value followed by a *println()*
  - *void println(char[] charArray)* – prints an array of characters followed by *println()*
  - *void println(double doubleValue)* – prints a double value followed by *println()* Methods
  - *void println(float floatValue)* – prints a float value followed by *println()*
  - *void println(int intValue)* – prints an int value followed by *println()*
  - *void println(long longValue)* – prints a long value followed by *println()*
  - *void println(Object obj)* – prints the value of the specified object's toString( ) method followed by println()
  - *void println(String string)* – prints a string followed by a line separator
  - *protected void SetError( )* – modifies the error flag to a value of "true"

4.4 *Langkah-langkah Praktikum*

4.4.1 *Baca Karakter*

1. Buatlah Project dan class baru dengan nama BacaKarakter.
2. Tulislah *source code* berikut:

```
import java.io.*;

class BacaKarakter{
    public static void main(String[] args){
        char c;
        try{
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Masukkan karakter (akhiri dengan \"q\") : ");
            do {
                c = (char) br.read();
                System.out.println("Karakter terbaca : "+c);
            } while (c!='q');
        } catch(IOException e){
            System.out.println("Ada error IO");
            System.exit(0);
        }
    }
}
```

3. Jalankan program tersebut!
4. Masukkan karakter ke dalam inputan.
5. Masukkan karakter ‘q’ untuk berhenti.

4.4.2 Baca String

1. Buatlah class baru dengan nama BacaString.
2. Tulislah *source code* berikut:

```
import java.io.*;

class BacaString{
    public static void main(String[] args){
        String str;
        try{
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Masukkan string (akhiri dengan \"end\") : ");
            do {
                str = br.readLine();
                System.out.println("Kata terbaca : "+str);
            } while (str.equalsIgnoreCase("end")==false);
        } catch(IOException e){
            System.out.println("Ada error IO");
            System.exit(0);
        }
    }
}
```

3. Jalankan program tersebut!
4. Masukkan kata dan kalimat dalam program tersebut!

4.4.3 *Baca File*

1. Buatlah class dengan nama cobaFile4!
2. Tulislah *source code* berikut!

```
import java.io.*;
class cobaFile4{
	public static void main(String[] args) {
		if(args.length != 1){
			System.out.println("Usage : java cobaFile4 <filetoread>");
			System.exit(1);
		}
		try{
			FileReader f = new FileReader(args[0]);
			BufferedReader r = new BufferedReader(f);
			String s = null;
			while((s=r.readLine())!=null){
				System.out.println(s);
			}
			r.close();
			f.close();
		} catch(FileNotFoundException e){
			System.out.println("File not found!");
			System.exit(1);
		} catch(IOException e){
			System.out.println("IO Error!");
			System.exit(1);
		}
	}
}
```

3. Kompile aplikasi tersebut menjari jar!
4. Jalankan aplikasi tersebut di command line, kemudian berilah argument telah file yang akan anda baca!

4.5 *Tugas*

1. Buatlah program untuk menulis teks ke dalam sebuah file text!
2. Buatlah program untuk membaca file tersebut, keluarkan isi text yang ada dalam file tersebut! Keluarkan jumlah huruf a, i, u, e, dan o yang ada dalam text tersebut!